

Surfando – An Activity Based Music Recommendation System

MUMT 621 (Music Information Acquisition, Preservation, and Retrieval) Final Project

By: Zored Ahmer

Description and Goal

This project was developed to provide a browser extension that took advantage of daily user internet activity to improve music recommendations. During configuration, users create playlists and for each playlist, they specify URLs, seed songs and optional, tuneable parameters. The URLs determine which websites the playlist should be active on. The seed songs (given as 'open.spotify' urls) are passed to the Spotify web api which uses them to generate a list of recommended songs that it returns. The tuneable parameters can be used to filter the entire set of recommendations to a more appropriate subset if desired (these parameters are used by Spotify's recommendation engine; this add-on has no control over how they are applied).

Due to limitations of the Spotify API, as well as developer knowledge (the developer has never made a browser add-on before), the add-on does not contain all the features originally proposed, but it does function, albeit with constraints and with an involved setup process. The rest of this document covers these constraints, the setup of the extension, known issues and bugs with the extension and finally some closing remarks. The Project is available on [Github](https://github.com/ZoredA/surfando)¹.

Limitations and Constraints

Playtime restricted to 30 seconds

The Spotify Web API allows developers to find information on the Spotify catalogue, but it does not provide a complete track playback url. This is a known problem with the platform and Spotify developers have said they wish to add the feature, but due to concerns over content security, they have not been able to do so [yet](#) (Web playback of Full Tracks · Issue #57 · spotify/web-api · GitHub 2017). The API provides a sample track for most songs and the add-on in its current form uses this sample track to playback music.

It may be possible to play the entire track on the Spotify web-player by opening a background tab with it. However, then a significant amount of further work and reverse-engineering the Spotify web player might be required so the extension can retain control over music playback. It may be possible to emulate this behavior by closing and opening tabs using the song length as a timer.

The current implementation does not support full play-back only 30 second clips.

Background Authentication Server Required

As part of the Spotify user authorization process, Spotify redirects the user to a web address specified by the application developer. The flow is meant to be 'app' -> 'Spotify service where user logs in' -> 'app', but a standalone application not accepting HTTP requests cannot be redirected to.

¹ <https://github.com/ZoredA/surfando>

That is why for this extension, a background server is run in conjunction (in an external command line window). The server binds to localhost and serves two important functions. A user accessing the server can login to Spotify and note their `renewal_token`. This token is a setting in Surfando settings page and is used by the application to get an `authorization_token` which is used for every request made to the Recommendations API. To renew the token, the web app contacts the localhost server which in turn contacts Spotify. That is why the server needs to run while the application is used.

It may be possible to not use a server and use the web extension to intercept Spotify's redirects, but we haven't tried yet and doing so may void Spotify security protocol.

No recognition of audio playback websites

The add-on does not contain a blacklist of websites it should stop playing music for (such as YouTube or SoundCloud) nor can it automatically detect other streams and stop accordingly. These features are likely implementable, but due to time constraints were not implemented.

Setup

There is some setup that needs to be done before the extension can be used.

Checkout the code and Install node.js

The surfando codebase is available on [Github](https://github.com/ZoredA/surfando)² and needs to be checked out. Clone the code, then run 'git submodule init', 'git submodule update' to retrieve a submodule library.

While any server conforming to the required renew token request can be used, the add-on repository includes a node.js sample login application provided by Spotify³. The included application is identical to the Spotify original, but with one line change to allow for cross domain web requests.

After checking out the code with Git, install [node.js](https://nodejs.org/en/)⁴ and make sure it can be used from the command line (this can be tested with a 'node -v' command to print the version).

Register an application

The Spotify API requires developers to register their applications ahead of time and generate client credentials that are used by the application. These credentials cannot be bundled with this extension as it is open source and anyone else can hijack them to get information on potential users. To register a Spotify extension, follow the instructions in the [tutorial](https://developer.spotify.com/web-api/tutorial/)⁵ (in the Registering Your Application section). Be sure to press the Save button after you are done entering the information.

After node.js has been installed and you have created client credentials for the app, navigate to "surfando/web-api-auth-examples/" and type 'npm install' to install dependencies. Then navigate to "surfando/web-api-auth-examples/authorization_code" and write the credentials in app.js as illustrated in Figure 2.

² <https://github.com/ZoredA/surfando>

³ <https://github.com/spotify/web-api-auth-examples>

⁴ <https://nodejs.org/en/>

⁵ <https://developer.spotify.com/web-api/tutorial/>

```

var express = require('express'); // Express web server framework
var request = require('request'); // "Request" library
var querystring = require('querystring');
var cookieParser = require('cookie-parser');

var client_id = 'SomeLetters'; // Your client id
var client_secret = 'SomeSecretLetters'; // Your secret
var redirect_uri = 'http://localhost:8888/callback'; // Your redirect uri

```

Figure 1 Credentials in source file: app.js

Open a web browser, and navigate to localhost:8888. Press Log in with Spotify, enter your Spotify credentials and you will be redirected back to a summary page (sample shown in Figure 2).

Then on the command line and in the authorization_code directory, run the command 'node app.js'. The server should start and bind to localhost on port 8888.

The refresh token is the token required by Surfando to operate. Do not copy it from the summary, rather copy the url. The url will be of the form "localhost:8888/#access_token=B...&refresh_token=....".

Write down the value of this refresh_token (ideally in a permanent text file in case Firefox erases Surfando's settings) and leave the server running (you may close the tab, just not the command line program). If you do close the node server, simply restart it with "node app.js". There is no need to generate another refresh_token.

Logged in as

Display name	
Id	{some id}
Email	{some email}
Spotify URI	https://open.spotify.com/user/{userid}
Link	https://api.spotify.com/v1/users/{userid}
Profile Image	
Country	CA

oAuth info

Access token BASDJHKLJASHDLJASDHASLDHASLJDLJHDAS...

Refresh token ASDLKJASLDHLAJSDHASL...

Obtain new token using the refresh token

Figure 2 The Summary Page with fake data. You will want the Refresh Token from the web address displayed in your browser

Install the add-on

Open Mozilla Firefox and enter 'about:debugging' into the toolbar. Press Load Temporary Add-on and select any file in the root surfando directory. Complete instructions as well as a video can be found at the Mozilla [documentation](#)⁶. Note: Much of the extension API is cross-platform so the add-on might work in Chrome but it has not been tested at all.

Create Initial Settings

The add-on cannot function out of the box and must be configured. Navigate to "about:addons" (or use the Firefox menu and click Add-ons). Press the options button on Surfando. The window should now display a large form.

Write the refresh token previous generated in the Refresh Token field. (Note: write just the token and not something like refresh_token='...' and without any apostrophes.) The row below the refresh token field allows for switching between playlists and creating new playlists.

For each playlist enter as many urls as you would like using the add button. Each url can be individually removed or edited. Figure 3 illustrates this. Note that the add-on only looks at the subdomain and the domain. So docs.google.com is treated as a separate url from google.com but google.com/ is the same as google.com/doodles. The only seeds accepted are spotify links to **tracks** and **artists** and these links can be generated by opening a song in the web or desktop player and sharing a link. For example, <https://open.spotify.com/track/77NNZQSqzLNqh2A9JhLRkg> is the url for Journey's Don't Stop Believin'.

The screenshot shows a web interface for configuring the Surfando add-on. At the top, there is a dropdown menu with 'work' selected, a minus button, a 'New Playlist' button, and a plus button. Below this, the 'work Urls' section contains a text input field with an 'ADD' button, followed by two rows of text inputs containing 'https://docs.google.cc' and 'https://developer.moz', each with a minus button. The 'work Seeds' section follows, with a text input field and an 'ADD' button, and one row with 'https://open.spotify.co' and a minus button.

Figure 3 Urls and Seeds

⁶ https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Temporary_Installation_in_Firefox

During actual playback, the seeds are used, and any songs played that are not skipped by the user get added to the seed collection. Once the seed collection hits five seeds, previous ones get cut off (the API only supports a maximum of five seeds)

All Tuneable Options are optional. Tuneable options are saved per playlist and this allows for finely tuned playlists.

Press the checkmark if you wish to include an option in your recommendations (note: these options filter results, so it is possible to retrieve no recommendations at all if too many are used or they are mixed in unusual ways).

Each slider represents a minimum and maximum range for that value. So, if you a user sets the Energy slider to 0.44 and 0.86, you want all songs with an Energy that falls in between those two values. If an option has no theoretical upper or lower bound, two input boxes are used instead (the first input box is the minimum and the second is the maximum). The current design of the system does not allow for one or the other. If you wish to enter a minimum value, you must enter a maximum too (but can set it arbitrarily high to make it non-important).

The last section of the page contains the Target options. These are the same tuneable options but rather than a minimum or maximum, these are values the Spotify recommendation engine will try to be close to. So, if a user sets a Tempo of 120, Spotify will center the returned recommendations to have a BPM of 120.

Information on these options and what they represent can be found on the Spotify [recommendation documentation](https://developer.spotify.com/web-api/get-recommendations/)⁷.


The image shows a user interface for configuring Spotify recommendations. It features several sections with sliders and checkboxes. The first section includes 'Acousticness (float)' with a slider from 0.00 to 1.00, 'Danceability (float)' with a slider from 0.24 to 0.96, and 'Duration (ms) (int)' with input boxes for 'Minimum Value' and 'Maximum Value'. The second section includes 'Energy (float)' with a slider from 0.00 to 1.00, 'Instrumentalness (float)' with a slider from 0.00 to 1.00, and 'Key (int)' with input boxes for 'Minimum Value' and 'Maximum Value'. Checkboxes are present next to each option name, indicating whether it is selected for recommendations.

Figure 4 Some Tuneable Options

After you are finished, press the save button. You can switch between playlists without losing data, but closing the tab without pressing save will result in the loss of any new settings.

Once the settings are saved, the application should re-load the settings automatically, but if it fails to function, navigate to about:debugging and press reload (this won't delete any saved settings).

⁷ <https://developer.spotify.com/web-api/get-recommendations/>

To use the add-on, press the Surfando icon () and press play. Figure 5 illustrates the surfando popup. The Play button starts a song, pause pauses it (you must press play to resume it), next moves to the next song (and in doing so omits the current song from the seed for the next recommendation call), previous moves back and Spotify opens a tab with the Spotify web player (where any members can listen to the complete version of the song).

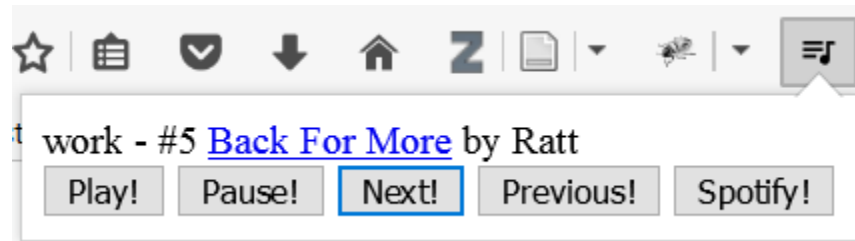


Figure 5 The Surfando popup

The default playlist the add-on first starts playing is determined by the order of the playlists in the dropdown. After spending two minutes on a website, Surfando checks if your current website is one of the websites specified in the Settings. If it is, it transitions to a different playlist (ideally with the next song but it might

Known Issues and Bugs

This section lists known issues and bugs that were not ironed out in time.

Insufficient Error Handling

There are many components interacting in this system (the popup, the background code, the settings page, the node.js server, and Spotify's services). A large amount of time was devoted trying to write error handlers, but the system is not robust enough to handle any error and report it to the user. Short of viewing the console log (which has a large amount of non-sequential output), debugging is likely be difficult. Some basic steps that should fix a lot of problems, however, are:

1. Ensure the settings page has a renew token saved
2. Ensure there are no badly entered settings (such as a float in an int field)
3. Turn off some tuneable settings (especially if you haven't tried them), they might be restricted the retrieved dataset to nothing.
4. Make sure the node js server is running and it can renew tokens (navigate to localhost:8888 and press the 'Obtain new token using the refresh token' button).
5. Reload the add-on in about:debugging
6. Press the play button

The Settings UI is not user friendly

The settings interface is functional, but it is not immediately clear what each setting represents and how to enter values for them. Ideally, it would include tool-tips and other useful information, but priority was given to getting it to work, so it has been submitted in an un-polished state. A few important settings that could improve the system but are missing now include a default playlist setting as well as the ability to pin seeds. Most importantly, the UI does not yet support genres, but the backend support is present, so this feature should not be too difficult too implement.

The popup is not updated in real time

The Surfando popup (Figure 5) displays information on the current song, but if closed, the information is no longer present when opened again. This is because the popup html is regenerated by Firefox every time the button is pressed. To see what song is being played press the Play! button or the Pause! button. The Spotify! button will still work even if the popup is not reflecting the current song.

Hypothetical Major Crash

During development, the add-on crashed Firefox (either via an infinite loop or more likely too large a call stack) three times, but only after a significant period of time. The exact cause of this bug was not discovered, but we suspect it is to do with the automatic renewal of Spotify tokens. That feature has been disabled. If you keep hearing the same song over and over, try reloading the add-on.

Until you are confident in the add-on do not use Firefox for important risk-averse activities (e.g. Google Doc might be fine because of its auto-save feature but typing a large amount into a pastebin post or a large forum post might be risky).

Closing Remarks

We developed an add-on that using pre-set playlists and current user activity to dynamically play music recommended by Spotify based on most recently played music (that the user did not skip). There are limitations, the greatest one being limited to 30 seconds of playback, but overall, we feel we've accomplished the original project goal of making an add-on that auto-plays music per your internet activity.

References

"Web playback of Full Tracks · Issue #57 · spotify/web-api · GitHub." Github. Accessed May 05, 2017, <https://github.com/spotify/web-api/issues/57>.